

Using The 8251 Universal Synchronous/Asynchronous Receiver/ Transmitter

Lionel Smith
Microcomputer Applications

Using the 8251 Universal Synchronous/ Asynchronous Receiver/Transmitter

Contents

INTRODUCTION

COMMUNICATION FORMATS

BLOCK DIAGRAM

- Receiver
- Transmitter
- Modem Control
- I/O Control

INTERFACE SIGNALS

- CPU-Related Signals
- Device-Related Signals

MODE SELECTION

PROCESSOR DATA LINK

CONCLUSION

APPENDIX A

- 8251 Design Hints

INTRODUCTION

The Intel 8251 is a Universal Synchronous/Asynchronous Receiver/Transmitter (USART) which is capable of operating with a wide variety of serial communication formats. Since many peripheral devices are available with serial interfaces, the 8251 can be used to interface a microcomputer to a broad spectrum of peripherals, as well as to a serial communications channel. The 8251 is part of the MCS-80™ Microprocessor Family, and as such it is capable of interfacing to the 8080 system with a minimum of external hardware.

This application note describes the 8251 as a component and then explains its use in sample applications via several examples. A specific use of the 8251 to facilitate communication between two MCS-80 systems is discussed in detail from both the hardware and software viewpoints. The first two sections of this application note describe the 8251 first from a functional standpoint and then on a detailed level. The function of each input and output pin is fully defined. The next section describes the various operating modes and how they can be selected, and finally, a sample design is discussed using the 8251 as a data link between the MCS-80 systems.

COMMUNICATION FORMATS

Serial communications, either on a data link or with a local peripheral, occurs in one of two basic formats; asynchronous or synchronous. These formats are similar in that they both require framing information to be added to the data to enable proper detection of the character at the receiving end. The major difference between the two formats is that the asynchronous format requires framing information to be added to each character, while the synchronous format adds framing information to blocks of data, or messages. Since the synchronous format is more efficient than the asynchronous format but requires more complex decoding, it is typically found on high-speed data links, while the asynchronous format is used on lower speed lines.

The asynchronous format starts with the basic data bits to be transmitted and adds a "START" bit to the front of them and one or more "STOP" bits behind them as they are transmitted. The START bit is a logical zero, or SPACE, and is defined as the positive voltage level by RS-232-C. The STOP bit is a logical one, or MARK, and is defined as the negative voltage level by RS-232-C. In current loop applications current flow normally indicates a MARK and lack of current a SPACE. The START bit tells the receiver to start assembling a character and allows the receiver to synchronize itself with the transmitter. Since this synchronization only

has to last for the duration of the character (the next character will contain a new START bit), this method works quite well assuming a properly designed receiver. One or more STOP bits are added to the end of the character to ensure that the START bit of the next character will cause a transition on the communication line and to give the receiver time to "catch up" with the transmitter if its basic clock happens to be running slightly slower than that of the transmitter. If, on the other hand, the receiver clock happens to be running slightly faster than the transmitter clock, the receiver will perceive gaps between characters but will still correctly decode the data. Because of this tolerance to minor frequency deviations, it is not necessary that the transmitter and receiver clocks be locked to the identical frequency for successful asynchronous communication.

The synchronous format, instead of adding bits to each character, groups characters into records and adds framing characters to the record. The framing characters are generally known as SYN characters and are used by the receiver to determine where the character boundaries are in a string of bits. Since synchronization must be held over a fairly long stream of data, bit synchronization is normally either extracted from the communication channel by the modem or supplied from an external source.

An example of the synchronous and asynchronous formats is shown in Figure 1. The synchronous format shown is fairly typical in that it requires two SYN characters at the start of the message. The asynchronous format, also typical, requires a START bit preceding each character and a single STOP bit following it. In both cases, two 8-bit characters are to be transmitted. In the asynchronous mode $10 \times n$ bits are used to transmit n characters and in the synchronous mode $8N + 16$ bits are used. For the example shown the asynchronous mode is actually more efficient, using 20 bits versus 32. To transmit a thousand characters in the asynchronous mode, however, takes 10,000 bits versus 8,016 for the synchronous format mode. For long messages the synchronous format becomes much more efficient than the asynchronous format; the crossover point for the examples shown in Figure 1 is eight characters, for which both formats require 80 bits.

In addition to the differences in format between synchronous and asynchronous communication, there are differences with regards to the type of modems that can be used. Asynchronous modems typically employ FSK (Frequency Shift Keying) techniques which simply generate one audio tone for a MARK and another for a SPACE. The receiving modem detects these tones on the telephone

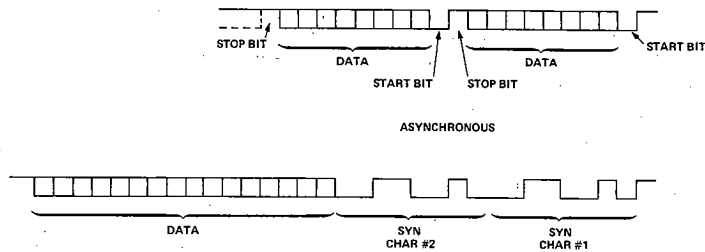


Figure 1. Transmission Formats

line, converts them to logical signals, and presents them to the receiving terminal. Since the modem itself is not concerned with the transmission speed, it can handle baud rates from zero to its maximum speed. Synchronous modems, in contrast to asynchronous modems, supply timing information to the terminal and require data to be presented to them in synchronism with this timing information. Synchronous modems, because of this extra clocking, are only capable of operating at certain preset baud rates. The receiving modem, which has an oscillator running at the same frequency as the transmitting modem, phase locks its clock to that of the transmitter and interprets changes of phase as data.

In some cases it is desirable to operate in a hybrid mode which involves transmitting data with the asynchronous format using a synchronous modem. This occurs when an increase in operating speed is required without a change in the basic protocol of the system. This hybrid technique is known as isosynchronous and involves the generation of the start and stop bits associated with the asynchronous format, while still using the modem clock for bit synchronization.

The 8251 USART has been designed to meet a broad spectrum of requirements in the synchronous, asynchronous, and isosynchronous modes. In the synchronous mode the 8251 operates with 5, 6, 7, or 8-bit characters. Even or odd parity can be optionally appended and checked. Synchronization can be achieved either externally via added hardware or internally via SYN character detection. SYN detection can be based on one or two characters which may or may not be the same. The single or double SYN characters are inserted into the data stream automatically if the software fails to supply data in time. The automatic generation of SYN characters is required to prevent the loss of synchronization. In the asynchronous mode the 8251 operates with the same data and parity structures as it does in the synchronous mode. In addition to appending a START bit to this data, the

8251 appends 1, 1½, or 2 STOP bits. Proper framing is checked by the receiver and a status flag set if an error occurs. In the asynchronous mode the USART can be programmed to accept clock rates of 16 or 64 times the required baud rate. Isosynchronous operation is a special case of asynchronous with the multiplier rate programmed as one instead of 16 or 64. Note that X1 operation is only valid if the clocks of the receiver and transmitter are synchronized.

The 8251 USART can transmit the three formats in half or full duplex mode and is double-buffered internally (i.e., the software has a complete character time to respond to a service request). Although the 8251 supports basic data set control signals (e.g., DTR and RTS), it does not fully support the signaling described in EIA-RS-232-C. Examples of unsupported signals are Carrier Detect (CF), Ring Indicator (CE), and the secondary channel signals. In some cases an additional port will be required to implement these signals. The 8251 also does not interface to the voltage levels required by EIA-RS-232-C; drivers and receivers must be added to accomplish this interface.

BLOCK DIAGRAM

A block diagram of the 8251 is shown in Figure 2. As can be seen in the figure, the 8251 consists of five major sections which communicate with each other on an internal data bus. The five sections are the receiver, transmitter, modem control, read/write control, and I/O Buffer. In order to facilitate discussion, the I/O Buffer has been shown broken down into its three major subsections: the status buffer, the transmit data/command buffer, and the receive data buffer.

Receiver

The receiver accepts serial data on the RxD pin and converts it to parallel data according to the appropriate format. When the 8251 is in the asynchronous mode and it is ready to accept a character

APPLICATIONS

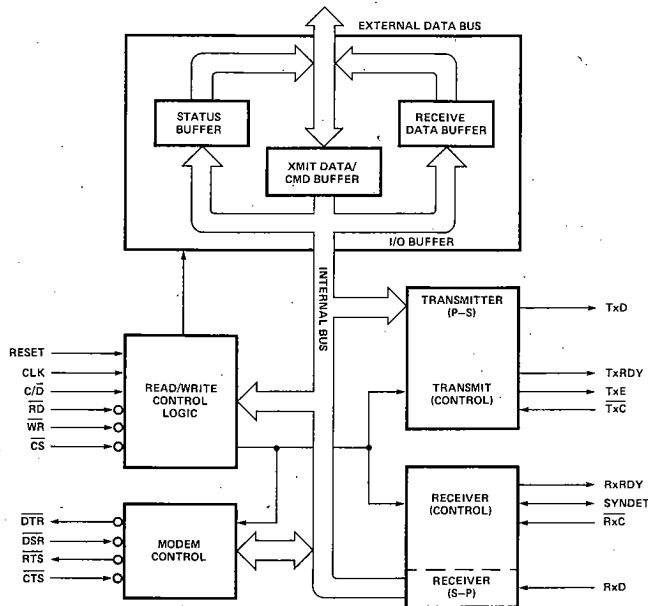


Figure 2. 8251 Block Diagram

(i.e., it is not in the process of receiving a character), it looks for a low level on the RxD line. When it sees the low level, it assumes that it is a START bit and enables an internal counter. At a count equivalent to one-half of a bit time, the RxD line is sampled again. If the line is still low, a valid START bit has probably been received and the 8251 proceeds to assemble the character. If the RxD line is high when it is sampled, then either a noise pulse has occurred on the line or the receiver has become enabled in the middle of the transmission of a character. In either case the receiver aborts its operation and prepares itself to accept a new character. After the successful reception of a START bit the 8251 clocks in the data, parity, and STOP bits, and then transfers the data on the internal data bus to the receive data register. When operating with less than 8 bits, the characters are right-justified. The RxRDY signal is asserted to indicate that a character is available.

In the synchronous mode the receiver simply clocks in the specified number of data bits and transfers them to the receiver buffer register, setting RxRDY. Since the receiver blindly groups data bits into characters, there must be a means of synchronizing the receiver to the transmitter so that the proper character boundaries are maintained in the serial data stream. This synchronization is achieved in the HUNT mode.

In the HUNT mode the 8251 shifts in data on the

RxD line one bit at a time. After each bit is received, the receiver register is compared to a register holding the SYN character (program loaded). If the two registers are not equal, the 8251 shifts in another bit and repeats the comparison. When the registers compare as equal, the 8251 ends the HUNT mode and raises the SYNDT line to indicate that it has achieved synchronization. If the USART has been programmed to operate with two SYN characters the process is as described above, except that two contiguous characters from the line must compare to the two stored SYN characters before synchronization is declared. Parity is not checked. If the USART has been programmed to accept external synchronization, the SYNDT pin is used as an input to synchronize the receiver. The timing necessary to do this is discussed in the SIGNALS section of this note. The USART enters the HUNT mode when it is initialized into the synchronous mode or when it is commanded to do so by the command instruction. Before the receiver is operated, it must be enabled by the RxR bit (D₂) of the command instructions. If this bit is not set the receiver will not assert the RxRDY bit.

Transmitter

The transmitter accepts parallel data from the processor, adds the appropriate framing information, serializes it, and transmits it on the TxD pin. In the asynchronous mode the transmitter always

adds a START bit; depending on how the unit is programmed, it also adds an optional even or odd parity bit, and either 1, 1½, or 2 STOP bits. In the synchronous mode no extra bits (other than parity, if enable) are generated by the transmitter unless the computer fails to send a character to the USART. If the USART is ready to transmit a character and a new character has not been supplied by the computer, the USART will transmit a SYN character. This is necessary since synchronous communications, unlike asynchronous communications, does not allow gaps between characters. If the USART is operating in the dual SYN mode, both SYN characters will be transmitted before the message can be resumed. The USART will not generate SYN characters until the software has supplied at least one character; i.e., the USART will fill 'holes' in the transmission but will not initiate transmission itself. The SYN characters which are to be transmitted by the USART are specified by the software during the initialization procedure. In either the synchronous or asynchronous modes, transmission is inhibited until TxEnable and the CTS input are asserted.

An additional feature of the transmitter is the ability to transmit a BREAK. A BREAK is a period of continuous SPACE on the communication line and is used in full duplex communication to interrupt the transmitting terminal. The 8251 USART will transmit a BREAK condition as long as bit 3 (SBRK) of the command register is set.

Modem Control

The modem control section provides for the generation of RTS and the reception of CTS. In addition, a general purpose output and a general purpose input are provided. The output is labeled DTR and the input is labeled DSR. DTR can be asserted by setting bit 2 of the command instruction; DSR can be sensed as bit 7 of the status register. Although the USART itself attaches no special significance to these signals, DTR (Data Terminal Ready) is normally assigned to the modem, indicating that the terminal is ready to communicate and DSR (Data Set Ready) is a signal from the modem indicating that it is ready for communications.

I/O Control

The Read/Write Control Logic decodes control signals on the 8080 control bus into signals which gate data on and off the USART's internal bus and controls the external I/O bus (DB₀–DB₇). The truth table for these operations is as follows:

If neither READ or WRITE is a zero, then the USART will not perform an I/O function. READ

CE	C/D	READ	WRITE	Function
0	0	0	1	CPU Reads Data from USART
0	1	0	1	CPU Reads Status from USART
0	0	1	0	CPU Writes Data to USART
0	1	1	0	CPU Writes Command to USART
1	X	X	X	USART Bus Floating (NO-OP)

and WRITE being a zero at the same time is an illegal state with undefined results. The Read/Write Control Logic contains synchronization circuits so that the READ and WRITE pulses can occur at any time with respect to the clock inputs to the USART.

The I/O buffer contains the STATUS buffer, the RECEIVE DATA buffer and the XMIT DATA/CMD buffer as shown in Figure 2. Note that although there are two registers which store data for transfer to the CPU (STATUS and RECEIVE DATA), there is only one register which stores data being transferred to the USART. The sharing of the input register for both transmit data and commands makes it important to ensure that the USART does not have data stored in this register before sending a command to the device. The TxRDY signal can be monitored to accomplish this. Neither data nor commands should be transferred to the USART if TxRDY is low. Failure to perform this check can result in erroneous data being transmitted.

INTERFACE SIGNALS

The interface signals of the 8251 USART can be broken down into two groups – a CPU-related group and a device-related group. The CPU-related signals have been designed to optimize the attachment of the 8251 to a MCS-80™ system. The device-related signals are intended to interface a modem or like device. Since many peripherals (TTY, CRT, etc.) can be obtained with a modem-like interface, the USART has a broad range of applications which do not include a modem. Note that although the USART provides a logical interface to an EIA-RS-232 device, it does not provide EIA compatible drive, and this must be added via circuitry external to the 8251. As an example of a peripheral interface application and to aid in understanding the signal descriptions which follow, Figure 3 shows a system configured to interface with a TTY or CRT.

APPLICATIONS

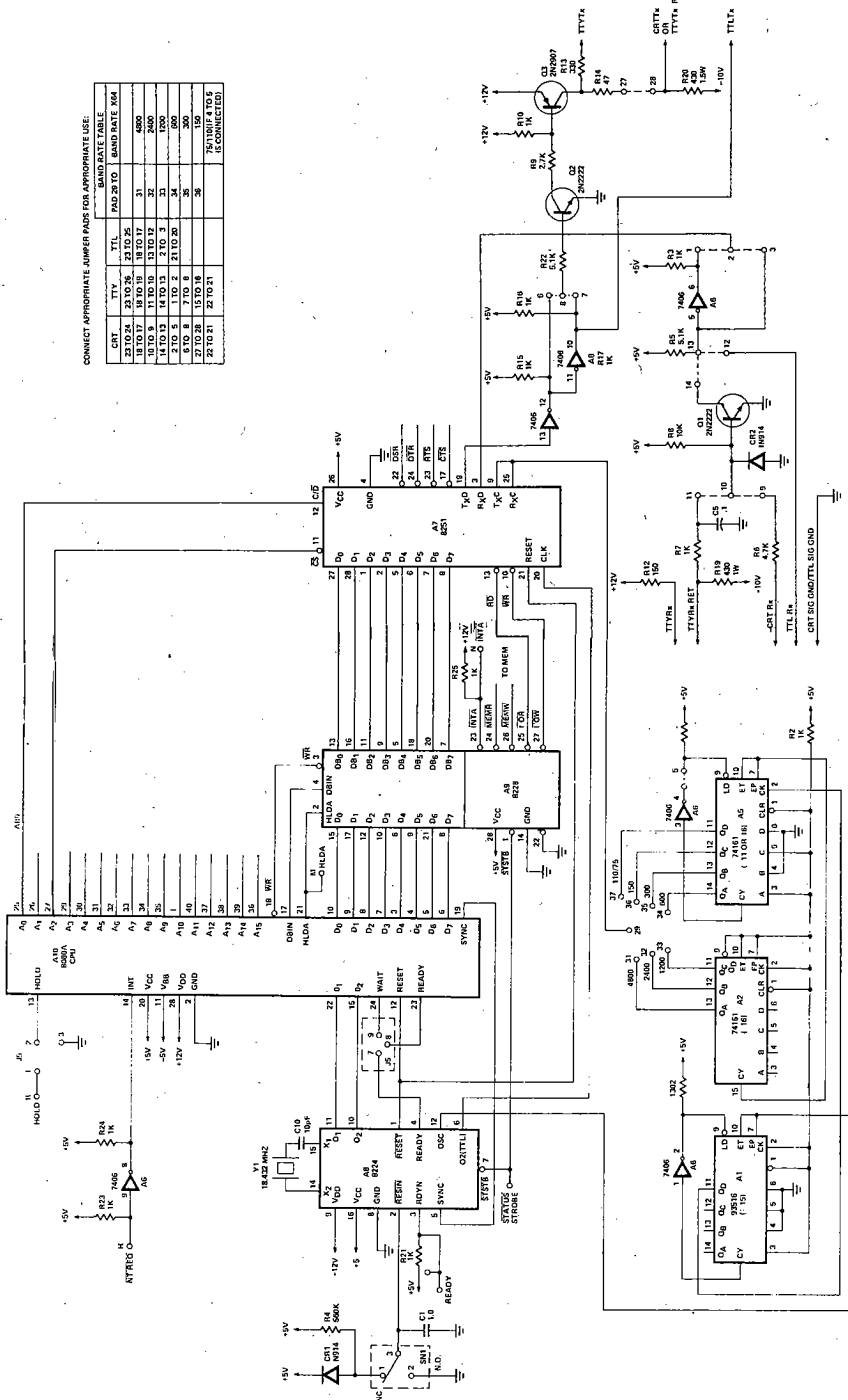


Figure 3. Terminal Interface

APPLICATIONS

CPU-Related Signals

V _{CC} (26)	I	+5 Volt Supply					status or data onto the data bus.
GND (4)	I	+5 Volt Common			\overline{WR} (10)	I	A low on this input causes the USART to accept data on the data bus as either a command or as a data character.
CLK (20)	I	The CLK input generates internal device timing. No external inputs or outputs are referenced to CLK, but the frequency of CLK must be greater than 30 times the Receiver or Transmitter clock inputs for synchronous mode or 4.5 times the clock inputs for an asynchronous mode. An additional constraint is imposed by the electrical specifications (ref. Appendix B) which require the period of CLK be between 0.42 μ sec and 1.35 μ sec. The CLK input can generally be connected to the Phase 2 (TTL) output of the 8224 clock generator.			TxRDY (15)	O	<i>Transmitter Ready.</i> This output signals the CPU that the USART is ready to accept a data character or command. It can be used as an interrupt to the system or, for polled operation, the CPU can check TxRDY using the status read operation. Note, however, that while the TxRDY status bit will be asserted whenever the XMIT DATA/CMD buffer is empty, the TxRDY output will be asserted only if the buffer is empty and the USART is enabled to transmit (i.e., CTS is low and TxEN is high). TxRDY will be reset when the USART receives a character from the program.
RESET (21)	I	A high on this input performs a master reset on the 8251. The device returns to the idle mode and will remain there until reinitialized with the appropriate control words.			TxE (18)	O	<i>Transmitter Empty.</i> A high output on this line indicates that the parallel to serial converter in the transmitter is empty. In the synchronous mode, if the CPU has failed to load a new character in time, TxE will go high momentarily as SYN characters are loaded into the transmitter to fill the gap in transmission.
DB ₇ -DB ₀ (8,7,6,5,2,1, 28,27)	I/O	The DB signals form a three-state bus which can be connected to the CPU data bus. Control, status, and data are transferred on this bus. Note that the CPU always remains in control of the bus and all transfers are initiated by it.			RxRDY (14)	O	<i>Transmitter Ready.</i> This output goes high to indicate that the 8251 has received a character on its serial input and is ready to transfer it to the CPU. Although the receiver runs continuously, RxRDY will only be asserted if the RxE (Receive Enable) bit in the command register has been set. RxRDY can be connected to the interrupt structure or, for polled operation, the CPU can check the condition of RxRDY using a status read operation. RxRDY will be reset when the character is read by the CPU.
\overline{CS} (11)	I	<i>Chip Select.</i> A low on this input enables communication between the USART and the CPU. Chip Select should go low when the USART is being addressed by the CPU.					
C/ \overline{D} (12)	I	<i>Control/Data.</i> During a read operation this pin selects either status or data to be input to the CPU (high=status, low=data). During a write operation this pin causes the USART to interpret the data on the bus as a command if it is high or as data if it is low.					
\overline{RD} (13)	I	A low on this input causes the USART to gate either					

APPLICATIONS

SYNDET (16) I/O *Synch Detect.* This line is used in the synchronous mode only. It can be either an input or output, depending on whether the initialization program sets the USART for external or internal synchronization. SYNDET is reset to a zero by RESET. When in the internal synchronization mode, the USART uses SYNDET as an output to indicate that the device has detected the required SYN character(s). A high output indicates synchronization has been achieved. If the USART is programmed to operate with double SYN characters, SYNDET will go high in the middle of the last bit of the second SYN character. SYNDET will be reset by a status read operation. When in the external synchronization mode a positive-going input on the SYNDET line will cause the 8251 to start assembling characters on the next falling edge of $\overline{\text{RxC}}$. The high input should be maintained at least for one $\overline{\text{RxC}}$ cycle following this edge.

Device-Related Signals

$\overline{\text{DTR}}$ (24) O *Data Terminal Ready.* This is a general purpose output signal which can be set low by programming a '1' in command instruction bit 1. This signal allows additional device control.

$\overline{\text{DSR}}$ (22) I *Data Set Ready.* This is a general purpose input signal. The status of this signal can be tested by the CPU through a status read. This pin can be used to test device status and is read as bit 7 of the status register.

$\overline{\text{RTS}}$ (23) O *Request to Send.* This is a general purpose output signal equivalent to $\overline{\text{DTR}}$. RTS is normally used to request that the modem prepare itself to transmit (i.e., establish carrier). RTS can be asserted

(brought low) by setting bit 5 in the command instruction.

$\overline{\text{CTS}}$ (17) I *Clear to Send.* A low on this input enables the USART to transmit data. CTS is normally generated by the modem in response to a $\overline{\text{RTS}}$.

$\overline{\text{RxC}}$ (25) I *Receiver Clock.* This clock controls the data rate of characters to be received by the USART. In the synchronous mode $\overline{\text{RxC}}$ is equivalent to the baud rate, and is supplied by the modem. In asynchronous mode $\overline{\text{RxC}}$ is 1, 16, or 64 times the baud rate. The clock division is preselected by the mode control instruction. Data is sampled by the USART on the rising edge of $\overline{\text{RxC}}$.

RxD (3) I *Receiver Data.* Characters are received serially on this pin and assembled into parallel characters. RxD is high true (i.e., High = MARK or ONE).

$\overline{\text{TxC}}$ (9) I *Transmitter Clock.* This clock controls the rate at which characters are transmitted by the USART. The relationship between clock rate and baud rate is the same as for $\overline{\text{RxC}}$. Data is shifted out of the USART on the falling edge of $\overline{\text{TxC}}$.

TxD (19) O *Transmit Data.* Parallel characters sent by the CPU are transmitted serially by the USART on this line. TxD is high true (i.e., High = MARK or ONE).

MODE SELECTION

The 8251 USART is capable of operating in a number of modes (e.g., synchronous or asynchronous). In order to keep the hardware as flexible as possible (both at the chip and end product level), these operating modes are selected via a series of control outputs to the USART. These mode control outputs must occur between the time the USART is reset and the time it is utilized for data transfer. Since the USART needs this information to structure its internal logic it is essential to complete the initialization before any attempts are made at data transfer (including reading status).

A flowchart of the initialization process appears in Figure 4. The first operation which must occur following a reset is the loading of the mode control

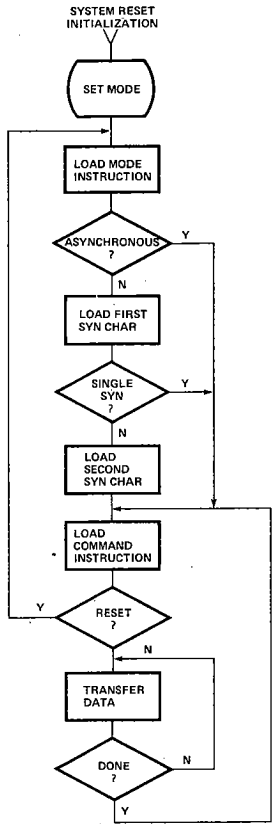


Figure 4. Initialization Flowchart

register. The mode control register is loaded by the first control output ($C/\bar{D}=1$, $\bar{RD}=1$, $WR=0$, $\bar{CS}=0$) following a reset. The format of the mode control instruction is shown in Figure 5. The instruction can be considered as four 2-bit fields. The first 2-bit field ($D_1 D_0$) determines whether the USART is to operate in the synchronous (00) or asynchronous mode. In the asynchronous mode this field also controls the clock scaling factor. As an example, if D_1 and D_0 are both ones, the RxC and TxC will be divided by 64 to establish the baud rate. The second field, D_3-D_2 , determines the number of data bits in the character and the third, D_5-D_4 , controls parity generation. Note that the parity bit (if enabled) is added to the data bits and is not considered as part of them when setting up the character length. As an example, standard ASCII transmission, which is seven data bits plus even parity, would be specified as:

X X 1 1 1 0 X X

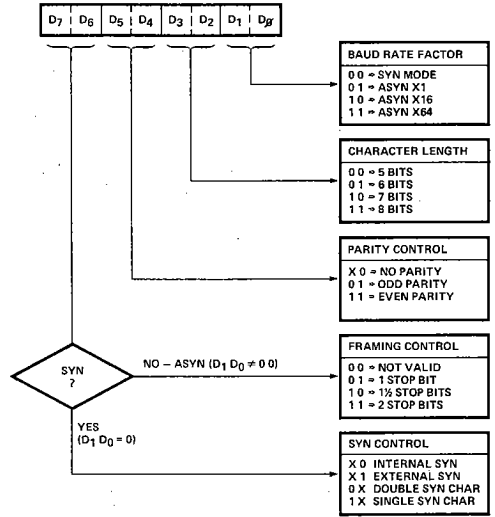


Figure 5. Mode Instruction Format

The last field, D_7-D_6 , has two meanings, depending on whether operation is to be in the synchronous or asynchronous mode. For the asynchronous mode (i.e., $D_1 D_0 \neq 00$), it controls the number of STOP bits to be transmitted with the character. Since the receiver will always operate with only one STOP bit, D_7 and D_6 only control the transmitter. In the synchronous mode ($D_1 D_0 = 00$), this field controls the synchronizing process. Note that the choice of single or double SYN characters is independent of the choice of internal or external synchronization. This is because even though the receiver may operate with external synchronization logic, the transmitter must still know whether to send one or two SYN characters should the CPU fail to supply a character in time.

Following the loading of the mode instruction the appropriate SYN character (or characters) must be loaded if synchronous mode has been specified. The SYN character(s) are loaded by the same control output instruction used to load the mode instruction. The USART determines from the mode instruction whether no, one, or two SYN characters are required and uses the control output to load SYN characters until the required number are loaded.

At completion of the load of SYN characters (or after the mode instruction in the asynchronous mode), a command character is issued to the USART. The command instruction controls the operation of the USART within the basic framework established by the mode instruction. The format of the command instruction is shown in

APPLICATIONS

Figure 6. Note that if, as an example, the USART is waiting for a SYN character load and instead is issued an internal reset command, it will accept the command as a SYN character instead of resetting. This situation, which should only occur if two independent programs control the USART, can be avoided by outputting three all zero characters as commands before issuing the internal reset command. The USART indicates its state in a status register which can be read under program control. The format of the status register read is shown in Figure 7.

When operating the receiver it is important to realize that Rx \bar{E} (bit 2 of the command instruction) only inhibits the assertion of RxRDY; it does not inhibit the actual reception of characters. Because the receiver is constantly running, it is possible for it to contain extraneous data when it is enabled. To avoid problems this data should be read from the USART and discarded. The read should be done immediately following the setting of Receive Enable in the asynchronous mode, and following the setting of Enter Hunt in the synchronous mode. It is not necessary to wait for RxRDY before executing the dummy read.

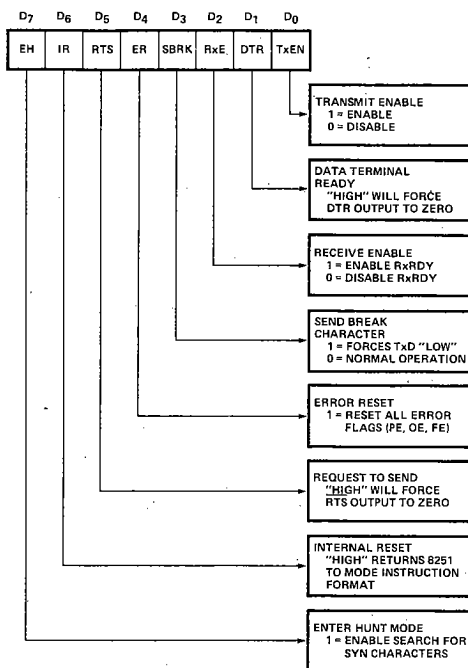


Figure 6. Command Instruction Format

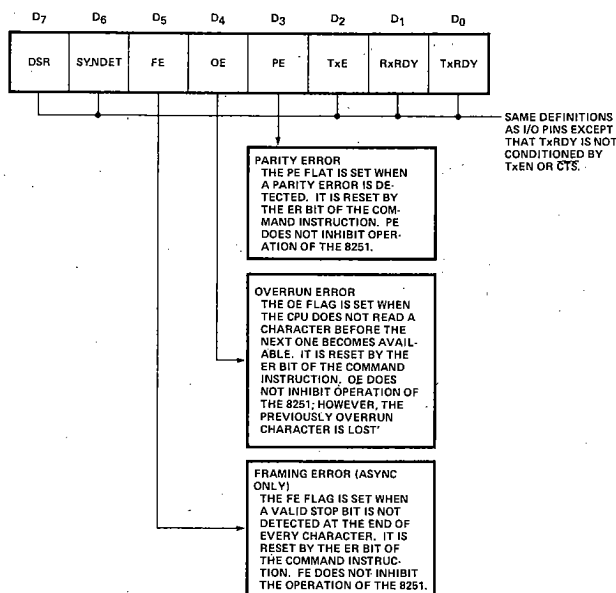


Figure 7. Status Register Format

PROCESSOR DATA LINK

The ability to change the operating mode of the USART by software makes the 8251 an ideal device to use to implement a serial communication link. A terminal initially configured with a simple asynchronous protocol can be upgraded to a synchronous protocol such as IBM Binary Synchronous Communication by a software only upgrade. In order to demonstrate the use of the 8251 USART, the remainder of this document will describe the implementation of an interrupt-driven, full duplex communication link on the Intel MDSTTM system. With minor modifications, the program developed could be used on the Intel SBC-80/10TM OEM card, thus implementing a data link between the two systems. Such a facility can be used to down-load programs, run diagnostics, and maintain common data bases in multiprocessor systems.

The factors which must be considered in the design of such a link include the desired transmission rate and format, the error checking requirements, the desirability of full duplex operation, and the physical implementation of the link. The basic requirement of the system described here is that it allow an Intel SBC-80/10 OEM card to be loaded from an MDS development system, either locally or on the switched telephone network. An additional constraint is that the modem used on the switched network be readily available and inexpensive. These requirements led to the choice of a modem such as the Bell 103A to implement the link. These modems, which support full duplex communication at up to 300 baud, are readily available from a number of sources at reasonable cost. These modems are also available in acoustically coupled versions which do not require permanent installation on the telephone network. Interface to the 103A modem is accomplished with nine wires: Protective Ground, Signal Ground, Transmitted Data, Received Data, Clear to Send, Data Set Ready, Data Terminal Ready, Carrier Detector, and Ringing Indicator.

The utilization of the interface signals to the modem is as follows:

Protective Ground	Protective Ground is used to bond the chassis ground of the modem to that of the terminal.
Signal Ground	Signal Ground provides a common ground reference between the modem and the terminal.
Transmitted Data	Transmitted Data is used to transfer serial data from the terminal to the modem.

Received Data	Received Data is used to transfer serial data from the modem to the terminal.
Clear to Send	Clear to Send indicates that the modem has established a connection with a remote modem and is ready to transmit data.
Data Set Ready	Data Set Ready indicates that the modem is connected to the telephone line and is in the data mode.
Data Terminal Ready	Data Terminal Ready is a signal from the terminal which permits the modem to enter the data mode.
Carrier Detector	Carrier Detector is identical to Clear to Send in the 103 modem and will not be used in this interface.
Ringing Indicator	Ringing Indicator indicates that the modem is receiving a ringing signal from the telephone system. This signal will not be used in the interface, since it is possible for the terminal to assert Data Terminal Ready whenever it is ready for the modem to "answer the telephone". The modem uses Data Set Ready to indicate that it has answered the call.

A block diagram showing the connections between the MDS and the SBC-80/10 through the modems is shown in Figure 8. Figure 9 shows the portion of the MDS monitor board devoted to the USARTs and Figure 10 shows the equivalent section of the SBC-80/10 board. Note that several signals on the MDS do not have the proper EIA defined voltage levels, and for this reason the adapter shown in Figure 11 was added to the MDS. The 390 pF capacitor was added to the 1488 driver to bring the rise time within EIA imposed limits of 30 volts/ μ sec. In Figure 7 the signal labels within the MDS and SBC-80/10 blocks correspond to the labels on the schematics, the signal labels within the modem blocks correspond to EIA conventions, and the signal labels on the wires between the blocks are abbreviations for the English language names of the signals.

As an example of how the USART clocks can be generated, circuits A27, A16, and A15 of Figure 9 form a divider of the OSC signal. The OSC signal has a frequency of 18.432 MHz and is generated by the 8224 which generates system timing for the 8080A. The 18.432 MHz signal results in a state time of 488 ns versus the normal 500 ns for the 8080A. (This does not violate 8080A specifications.) The 18.432 MHz signal can be divided by

APPLICATIONS

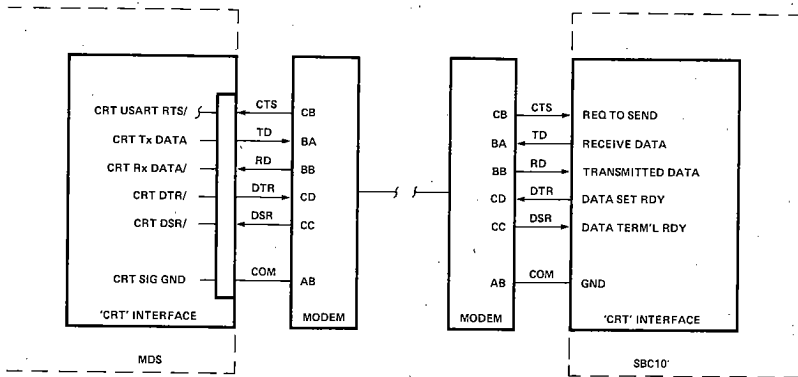


Figure 8. System Block Diagram

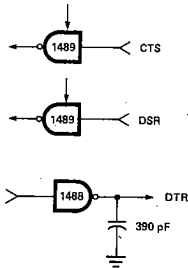


Figure 9. EIA Adapter

30 and then 64 to give a 9600 baud communication standard. The 9600 baud signal can be further divided to give 4800, 2400, 1200, 600, and 300 baud signals. The 1200 baud signal can be divided by 11 to give a 109.1 baud signal which is within 1% of the 110 baud standard signal rate. Note that because of constraints on the CLK input 9600 baud operation is not possible in the X64 mode. The divide by 64 can be accomplished by dividing by 4 with a counter and then 16 within the USART.

In order to keep the system as general purpose as possible, it was decided to transmit 8-bit data characters with an appended odd parity bit. Having a full 8-bit byte available for data enables the transmission of codes such as ASCII (which is 7-level with an additional parity bit) to be transmitted and received transparently in the system. Also, of course, it allows 8-bit bytes from the 8080A memory to be transferred in one transmission character. If error checking beyond the parity check is required, it could be added to the data record to be transmitted in the form of redundant check characters.

Before the software design of the system could be undertaken, it was necessary to decide whether service requests from the USART would be handled on a polled or interrupt driven mode. Polled operation normally results in more compact code but it requires that whatever programs are running concurrently with a transmission or reception must periodically either check the status of the USART or call a routine that does. Since it was not possible to determine what program might be running during a receive or transmit operation, it was decided to operate in an interrupt driven mode.

The program which operates the 8251 must be instructed as to what data it should transmit or receive from some other program resident in the 8080 system. To facilitate the discussion of the operation of the software, the following definitions will be made:

USRUN is the program which controls the operation of the 8251.

USER is a program which utilizes USRUN in order to effect a data transmission.

USER passes commands and parameters to USRUN by means of the control block shown in Figure 12. The first byte of the block contains the command which USER wants USRUN to execute. Valid contents of this byte are "C" which causes USRUN to initialize itself and the 8251, "R" which causes the execution of the data input (or READ) operation, and "W" which causes a data output (WRITE) operation. The second byte of the control block is used by USRUN to inform USER of the status of the requested operation. The third and fourth bytes specify the starting address of a buffer set up by USER which contains the data for a transmit operation or which will be used by USRUN to store received data. The fifth and sixth bytes are concatenated to form a positive binary

6-126

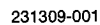


Figure 10. SBC 80/10 Serial I/O

APPLICATIONS

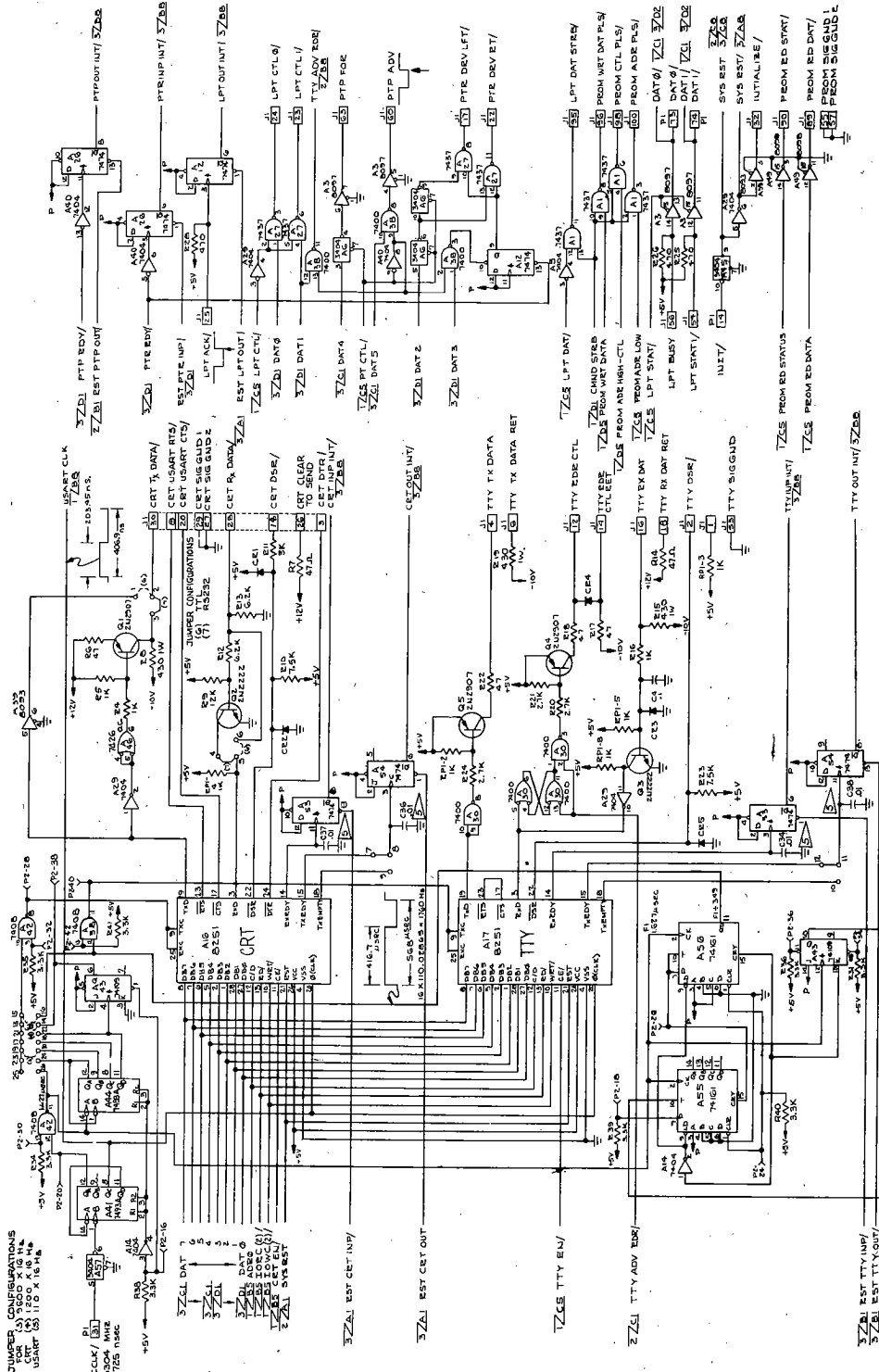


Figure 11. MDS Monitor Module

APPLICATIONS

number which specifies how many bytes of data USER wants transferred. The seventh and eighth bytes are concatenated and used by USRUN to count the number of bytes that have been transferred. When the required number of characters have been transferred, or if USRUN terminates a READ or WRITE due to an abnormal condition, then USRUN calls a subroutine at an address defined by the ninth and tenth bytes of the command block. This subroutine, which is provided by USER, must determine the state of the process and then take appropriate action.

Since USRUN must be capable of operation in a full duplex mode (i.e., be able to receive and transmit simultaneously), it keeps the address of two control blocks; one for a READ operation and one for a WRITE. The address of the controlling command block is kept in RAM locations labeled RCBA for the READ operation and TCBA for the WRITE operation. If RCBA (Receive Control Block Address) or TCBA (Transmit Control Block Address) is zero, it indicates that the corresponding operation is in an idle status.

Flowcharts of USRUN appear in Figure 13 and the listings appear in Figure 14. The first section of the flowcharts (Figures 13.1 and 13.2) consists of two subroutines which are used as convenient tools for operating on the control blocks. These routines are labeled LOADA and CLEAN. LOADA is entered with the address of a control block in registers H and L. Upon return registers D and E have been set equal to the address in the buffer which is the target of the next data transfer (i.e., $D,E = BAD + CCT$); and CCT (transferred byte count) has then been incremented. In addition, the B register is set to zero if the number of bytes that have been transferred is equal to the number requested (i.e., $CCT = RCT$). CLEAN, the second routine, is also entered with the address of a command block in the H and L registers. In addition, the Accumulator holds the status which will be placed in the STATUS byte of the command block. On exit the STATUS byte has been updated and the address of the completion routine has been placed in H and L.

Upon interrupt, control of the MCS-80 system is transferred to VECTOR (Figure 13.3). Vector is a program which saves the state of the system, gets the status of the USART and jumps to the RISR (Receive Interrupt Service Routine) or the TISR (Transmit Interrupt Service Routine), depending on which of the two ready flags is active. If neither ready flag is active, VECTOR restores the status of the running program, enables interrupts, and returns. (Interrupts are automatically disabled by the hardware upon an interrupt.) This exit from VECTOR, which is labeled VOUT, is used from other

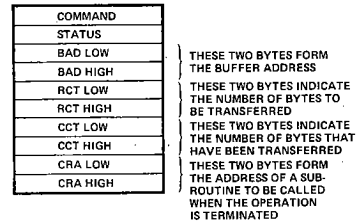


Figure 12. Control Block

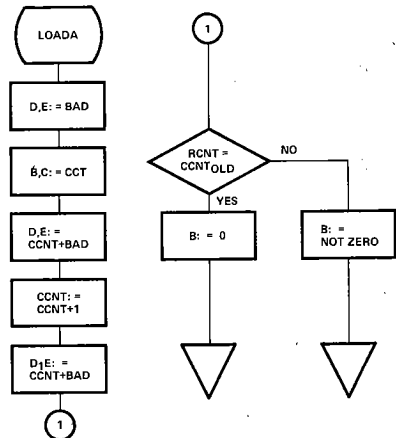


Figure 13.1. LOADA Subroutine

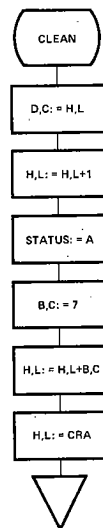


Figure 13.2. CLEAN Subroutine

APPLICATIONS

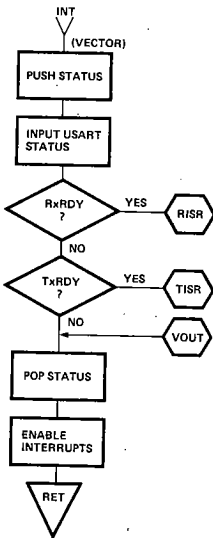


Figure 13.3. Interrupt Entry

portions of USRUN if return from the interrupt mode is required.

In addition to handling normal data transfers, TISR (Figure 13.4) checks a location in memory named TCMD in order to determine if the receive program wishes to send a command to the USART. Since the transmit data and command must share a buffer within the USART, any command output must occur when TxRDY is asserted. If TCMD is zero, TISR proceeds with the data transfer. If TCMD is non-zero, TISR calls TUTE (Transmit Utility, Figure 13.5) which, depending on the value

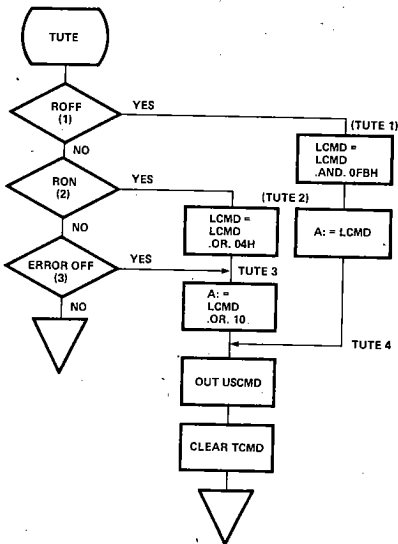


Figure 13.4. Transmit Interrupt Service Routine

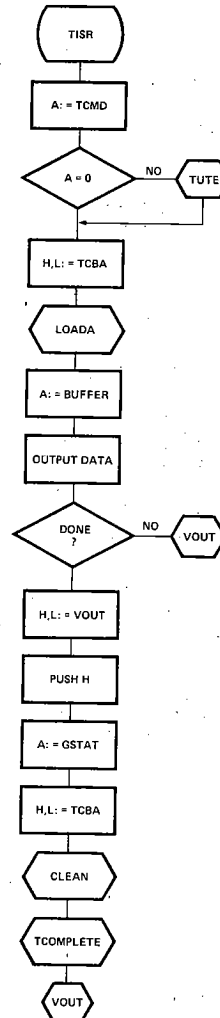


Figure 13.5. Transmit Utility Routine

in TCMD, turns off the receiver, turns on the receiver, or clears error conditions. Note that the error flags (parity, framing, and overrun) are always cleared by the software when the receiver is first enabled.

The flowchart of the RISR is shown in Figure 13.6. Note that in addition to terminating whenever the required number of characters have been received, the RISR also terminates if one of the error flags becomes set or if the received character matches a character found in a table pointed to by the label ETAB. This table, which starts at ETAB and continues until an all "ones" entry is found, can be used by USER to define special characters, such as EOT (End Of Transmission), which will terminate a READ operation. The remainder of Figure 13 (13.7) shows the decoding of the commands to USRUN. The listings also include a test USER which exercises USRUN. This program sets up a 256-byte transmit buffer and transfers it to a similar input buffer by means of a local loop. When both the READ and WRITE operations are complete, the test USER checks to insure that the two buffers are identical. If the buffers differ, the MDS monitor is called; if the data is correct, the test is repeated.

CONCLUSION

The 8251 USART has been described both as a device and as a component in a system. Since not only modems but also many peripheral devices have a serial interface, the 8251 is an extremely useful component in a microcomputer system. A particular advantage of the device is that it is capable of operating in various modes without requiring hardware modifications to the system of which it is a part. As with any complex subsystem, however, the 8251 USART must be carefully applied so that it can be utilized to full advantage in the overall system. It is hoped that this application note will aid in the designer in the application of the 8251 USART. As a further aid to the application of the 8251, the appendix of this document includes a list of design hints based on past experience with the 8251.

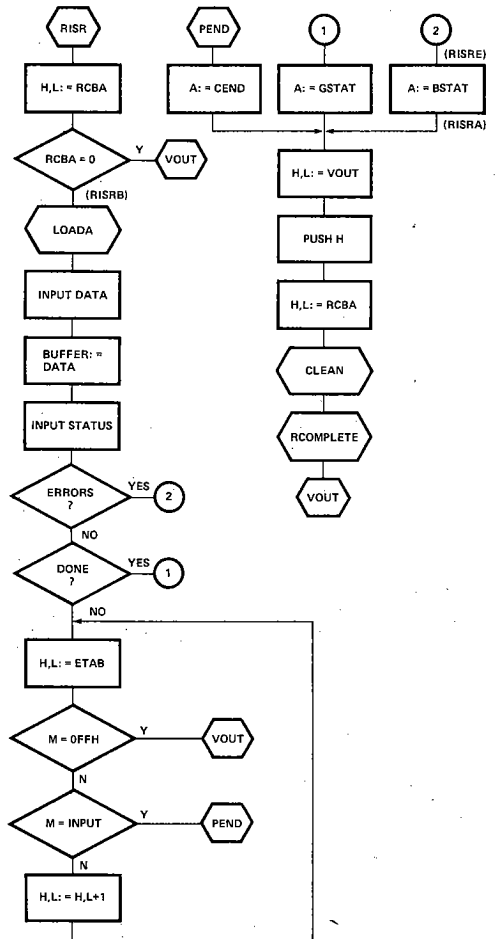


Figure 13.6. Receive Interrupt Service Routine

APPLICATIONS

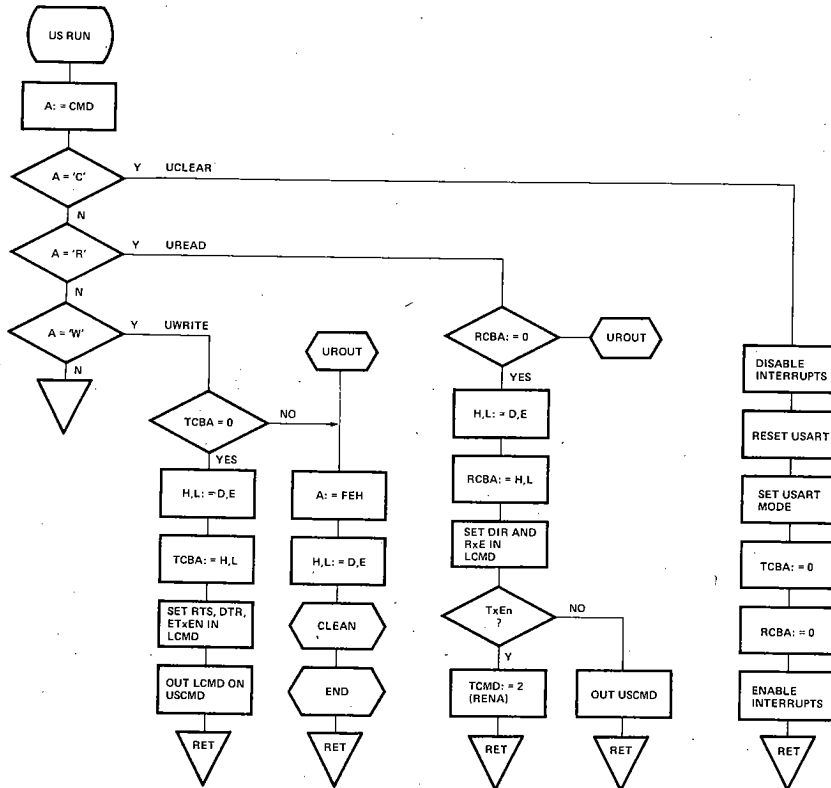


Figure 13.7. URUN Command Decode

APPLICATIONS

Figure 14. Program Listing

```

;*****
;
;      SYSTEM ORIGIN STATEMENT
;*****

4000          ORG      4000H

;*****
;
;      DATA STORAGE FOR TEST USER
;*****

4000          BUFIN:   DS      100H      ;INPUT BUFFER
4100          BUFOUT:  DS      100H      ;OUTPUT BUFFER
4200 5200      RBLOCK:  DB      'R',00H  ;RECEIVE CONTROL BLOCK
4202 0040      RBAD:    DW      BUFIN
4204 FF00      RRCT:    DW      OFFH
4206 0000      RCCT:    DW      00H
4208 1742      RCRA:    DW      RCR
420A 5700      TBLOCK:  DB      'W',00H  ;TRANSMIT CONTROL BLOCK
420C 0041      TBAD:    DW      BUFOUT
420E FF00      TRCT:    DW      OFFH
4210 0000      TCCT:    DW      00H
4212 2742      TCRA:    DW      TCR
4214 4300      GBLOCK:  DB      'C',00H
4216 00        FLAG:    DB      00H

;*****
;
;      COMPLETION ROUTINES
;*****

4217 AF        RCR:     XRA      A          ;CLEAR A
4218 323B42     STA      RCBA      ;TURN OFF RECEIVE
421B 323C42     STA      RCBA+1
421E 3A1642     LDA      FLAG      ;GET FLAG
4221 E60F        ANI      OFH      ;CLEAR UPPER FOUR BITS
4223 321642     STA      FLAG      ;RESTORE FLAG
4226 C9         RET
4227 AF        TCR:     XRA      A          ;CLEAR A
4228 323942     STA      TCBA      ;TURN OFF TRANSMIT
422B 323A42     STA      TCBA+1
422E 3A1642     LDA      FLAG      ;GET FLAG
4231 E6F0        ANI      OF0H     ;CLEAR LOWER FOUR BITS
4233 321642     STA      FLAG      ;RESTORE FLAG
4236 C9         RET      ;THEN RETURN

```

APPLICATIONS

```

;*****
;
;      SYSTEM EQUATES
;
;*****

```

```

00F5      USTAT   EQU      0F5H      ;USART STATUS ADDRESS
00F5      USCMD   EQU      0F5H      ;USART CMD ADDRESS
00F4      USDAI   EQU      0F4H      ;USART DATA INPUT ADDRESS
00F4      USDAO   EQU      0F4H      ;USART DATA OUTPUT ADDRESS
0000      GSTAT   EQU      00H       ;GQOD STATUS
00FF      BSTAT   EQU      0FFH      ;BAD STATUS
0001      CEND    EQU      01H

```

```

;*****
;
;      SYSTEM DATA TABLE
;
;*****

```

```

4237 00      LCMD:   DB      00H      ;CURRENT OPERATING COMMAND
4238 00      TCMD:   DB      00H      ;IF NON ZERO A COMMAND TO BE SENT
4239 0000     TCBA:   DW      00H      ;ADDRESS OF XMIT CBLOCK
423B 0000     RCBA:   DW      00H      ;ADDRESS OF RECEIVE CBLOCK
423D FF      MTAB:   DB      0FFH     ;END CHARACTER TABLE

```

APPLICATIONS

```
;*****
;
;      LOAD ADDRESS ROUTINE
;      LOADA IS ENTERED WITH THE ADDRESS OF A CONTROL
;      BLOCK IN H,L. ON EXIT D,E CONTAINS THE ADDRESS
;      WHICH IS THE TARGET OF THE NEXT DATA TRANSFER (BAD+CCNT)
;      AND B HAS BEEN SET TO ZERO IF THE REQUESTED NUMBER OF
;      TRANSFERS HAS BEEN ACCOMPLISHED. CCNT IS INCREMENTED
;      AFTER THE TARGET ADDRESS HAS BEEN CALCULATED.
;
;*****
```

```
423E 23      LOADA: INX      H      ;D,E GETS BUFFER ADDRESS
423F 23              INX      H
4240 5E              MOV      E,M
4241 23              INX      H
4242 56              MOV      D,M      ;DONE
4243 23              INX      H      ;B,C GETS COMPLETED COUNT (CCNT)
4244 23              INX      H
4245 23              INX      H
4246 4E              MOV      C,M
4247 23              INX      H
4248 46              MOV      B,M      ;DONE
4249 EB              XCHG              ;D,E GETS BAD+CCNT
424A 09              DAD      B
424B EB              XCHG              ;DONE
424C 03              INX      B      ;CCNT GETS INCREMENTED
424D 70              MOV      M,B
424E 2B              DCX      H
424F 71              MOV      M,C      ;DONE
4250 0B              DCX      B      ;DOES OLD CCNT=RCNT?
4251 2B              DCX      H
4252 7E              MOV      A,M
4253 90              SUB      B
4254 47              MOV      B,A
4255 C0              RNZ              ;NO-RETURN WITH B NOT ZERO
4256 2B              DCX      H
4257 7E              MOV      A,M
4258 91              SUB      C
4259 47              MOV      B,A
425A C9              RET              ;RETURN WITH B=0 IF RCNT=CCNT
```

APPLICATIONS

```

;*****
;
;      CLEAN-UP ROUTINE
;      CLEAN IS ENTERED WITH THE ADDRESS OF A CONTROL
;      BLOCK IN H,L AND A NEW STATUS TO BE
;      ENTERED INTO IT IN A. ON EXIT THE ADDRESS OF THE
;      CONTROL BLOCK IS IN D,E; THE STATUS OF THE BLOCK
;      HAS BEEN UPDATED; AND THE ADDRESS OF THE COMPLETION
;      ROUTINE IS IN H,L.
;*****

```

```

425B 5D      CLEAN:  MOV     E,L      ;SAVE THE ADDRESS OF THE COMMAND BLOCK
425C 54      MOV     D,H
425D 23      INX      H      ;POINT AT STATUS
425E 77      MOV     M,A      ;SET STATUS EQUAL TO A
425F 010700  LXI      B,7      ;SET INDEX TO SEVEN
4262 09      DAD     B      ;POINT AT COMPLETION ADDRESS
4263 7E      MOV     A,M      ;GET LOWER ADDRESS
4264 23      INX      H      ;POINT AT UPPER ADDRESS
4265 66      MOV     H,M      ;H GETS HIGH ADDRESS BYTE
4266 6F      MOV     L,A      ;L GETS LOW ADDRESS BYTE
4267 C9      RET

```

```

;*****
;
;      INTERRUPT VECTOR ROUTINE
;      VECTOR SAVES THE STATUS OF THE RUNNING PROGRAM
;      THEN READS THE STATUS OF THE USART TO DETERMINE
;      IF A RECEIVE OR TRANSMIT INTERRUPT OCCURRED.
;      VECTOR THEN CALLS THE APPROPRIATE SERVICE ROUTINE.
;      IF NEITHER INTERRUPTS OCCURRED THEN VECTOR RESTORES
;      THE STATUS OF THE RUNNING PROGRAM. THE SERVICE
;      ROUTINES USE THE EXIT CODE, LABELED VOUT, TO EFFECT
;      THEIR EXIT FROM INTERRUPT MODE.
;*****

```

```

4268 F5      VECTOR:  PUSH    PSW      ;PUSH STATUS INTO THE STACK
4269 C5      PUSH    B
426A D5      PUSH    D
426B E5      PUSH    H
426C DBF5    IN      USTAT  ;GET USART ADDRESS
426E DBFA    IN      OFAH   ;MDS-GET MONITOR CARD INT. STATUS
4270 0F      RRC      ;ROTATE TWO PLACES
4271 0F      RRC      ;SO THAT CARRY=RXRDY
4272 DA8842  JC      RISR   ;IF RXRDY GO TO SERVICE ROUTINE
4275 07      RLC      ;IF NOT ROTATE BACK
4276 07      RLC      ;LEAVING TXRDY IN CARRY
4277 DAD442  JC      TISR   ;IF TXRDY THEN GO TO SERVICE ROUTINE
427A 3EFC    MVI     A,0FCH  ;MDS-CLEAR OTHER LEVEL THREE INTERRUPTS
427C D3F3    OUT     0F3H   ;MDS
427E E1      VOUT:    POP     H      ;ELSE EXIT FROM INTERRUPT MODE
427F D1      POP     D
4280 C1      POP     B
4281 3E20    MVI     A,20H   ;MDS-RESTORE CURRENT LEVEL
4283 D3FD    OUT     0FDH   ;MDS
4286 FB      EI      ;ENABLE INTERRUPTS
4287 C9      RET

```

APPLICATIONS

;*****

;

;

;

;

;

;

;

;

;*****

RECEIVE INTERRUPT SERVICE ROUTINE;
RISR PROCESSES A RECEIVE INTERRUPT
AT THE END OF RECEIVE THE USER SUPPLIED
COMPLETION ROUTINE IS CALLED AND THEN AN
EXIT IS TAKEN THROUGH VOUT OF THE
VECTOR

4288	2A3B42	RISR:	LHLD	RCBA	
428B	3E82		MVI	A,82H	;MDS-CLEAR RECEIVE INTERRUPT
428D	D3F3		OUT	OF3H	;MDS
428F	2C		INR	L	
4290	2D		DCR	L	
4291	C29942		JNZ	RISRB	
4294	24		INR	H	
4295	25		DCR	H	
4296	CA7E42		JZ	VOUT	
4299	CD3E42	RISRB:	CALL	LOADA	;READY-SET UP ADDRESS
429C	DBF4		IN	USDAI	;GET INPUT DATA
429E	12		STAX	D	;AND PUT IN THE BUFFER
429F	4F		MOV	C,A	;SAVE INPUT DATA IN C
42A0	DBF5		IN	USTAT	;GET STATUS AGAIN
42A2	E638		ANI	38H	;MASK FOR ERROR FIELD
42A4	C2B942		JNZ	RISRE	;NOT ZERO-TAKE ERROR EXIT
42A7	04		INR	B	;B WAS 00 IF DONE
42A8	05		DCR	B	
42A9	C2BE42		JNZ	EXCHAR	;NOT DONE-EXIT
42AC	3E00		MVI	A,GSTAT	;A GETS GOOD STATUS
42AE	217E42	RISRA:	LXI	H,VOUT	;GET RETURN ADDRESS
42B1	E5		PUSH	H	;AND PUSH IT INTO THE STACK
42B2	2A3B42		LHLD	RCBA	;POINT H,L AT THE CMD BLOCK
42B5	CD5B42		CALL	CLEAN	;CALL CLEANUP ROUTINE
42B8	E9		PCHL		;EFFECTIVELY CALLS COMPLETION ROUTINE
					;RETURN IS TO VOUT BECAUSE OF PUSH H
42B9	3EFF	RISRE:	MVI	A,BSTAT	;A GETS BAD STATUS
42BB	C3AE42		JMP	RISRA	;OTHERWISE EXIT IS NORMAL
42BE	213D42	EXCHAR:	LXI	H,MTAB	;TEST CHARACTER AGAINST EXIT TABLE
42C1	7E	EXA:	MOV	A,M	
42C2	FEFF		CPI	OFFH	;END OF TABLE
42C4	CA7E42		JZ	VOUT	
42C7	B9		CMP	C	
42C8	CACF42		JZ	PEND	;MATCH-TERMINATE READ
42CB	23		INX	H	
42CC	C3C142		JMP	EXA	
42CF	3E01	PEND:	MVI	A,CEND	
42D1	C3AE42		JMP	RISRA	

APPLICATIONS

;*****

;

;

;

;

;

;

;

;

;*****

TRANSMIT INTERRUPT SERVICE ROUTINE
TISR PROCESSES TRANSMITTER INTERRUPTS
WHEN THE END OF A TRANSMISSION IS
DETECTED THE USER SUPPLIED COMPLETION
ROUTINE IS CALLED AND THEN AN EXIT IS
TAKEN THROUGH VOUT OF VECTOR

42D4 3A3842	TISR:	LDA	TCMD	;GET POTENTIAL COMMAND
42D7 B7		ORA	A	;DESIGNATE ON IT
42D8 C40443		CNZ	TUTE	;DO UTILITY COMMAND
42DB 3E81		MVI	A,081H	;MDS-CLEAR XMIT INTERRUPTS
42DD D3F3		OUT	OF3H	;MDS
42DF 2A3942		LHLD	TCBA	
42E2 2C		INR	L	;MAKE SURE HAVE VALID CONTROL BLOCK
42E3 2D		DCR	L	
42E4 C2EC42		JNZ	TISRA	;GOOD
42E7 24		INR	H	
42E8 25		DCR	H	
42E9 CA7E42		JZ	VOUT	;NON VALID BLOCK (H,L=0)
42EC CD3E42	TISRA:	CALL	LOADA	;SET UP ADDRESS
42EF 1A		LDAX	D	;GET DATA FROM BUFFER
42F0 D3F4		OUT	USDAO	;AND OUTPUT IT
42F2 04		INR	B	;B WAS 00 IF DONE
42F3 05		DCR	B	
42F4 C27E42		JNZ	VOUT	;NOT DONE-EXIT FROM SERVICE ROUTINE
42F7 217E42		LXI	H,VOUT	;SET UP RETURN ADDRESS
42FA E5		PUSH	H	;AND PUSH IT INTO THE STACK
42FB 3E00		MVI	A,GSTAT	;A GETS GOOD STATUS
42FD 2A3942		LHLD	TCBA	;POINT H,L AT COMMAND BLOCK
4300 CD5B42		CALL	CLEAN	;CALL CLEANUP ROUTINE
4303 E9		PCHL		;CALL COMPLETION ROUTINE
				;RETURN WILL BE TO VOUT
4304 FE01	TUTE:	CPI	01	;RECEIVER OFF
4306 CA2443		JZ	TUTE1	
4309 FE02		CPI	02	;RECEIVER ON
430B CA1443		JZ	TUTE2	
430E FE03		CPI	03	;CLEAR ERRORS
4310 CA1C43		JZ	TUTE3	
4313 C9		RET		
4314 3A3742	TUTE2:	LDA	LCMD	
4317 F604		ORI	04	
4319 323742		STA	LCMD	
431C 3A3742	TUTE3:	LDA	LCMD	
431F F610		ORI	10H	
4321 D3F5	TUTE4:	OUT	USCMD	
4323 C9		RET		
4324 3A3742	TUTE1:	LDA	LCMD	
4327 E6FB		ANI	OFBH	
4329 323742		STA	LCMD	
432C C32143		JMP	TUTE4	

APPLICATIONS

```
;*****
;
;
;
;
;
;
;*****
```

```
USART COMMAND BLOCK INTERPRETER
USRUN IS CALLED BY USER WITH THE ADDRESS
OF THE COMMAND BLOCK IN H,L. USRUN EXAMINES
THE BLOCK AND INITIALIZES THE REQUESTED OPERATION
```

```
432F 1A      USRUN: LDAX    D      ;GET THE CMD FROM THE BLOCK
4330 FE43     CPI      'C'      ;IS IT A CLEAR COMMAND?
4332 CA4043    JZ       UCLEAR    ;YES GO TO CLEAR ROUTINE
4335 FE52     CPI      'R'      ;IS IT A READ COMMAND?
4337 CA5D43    JZ       UREAD     ;YES-GO TO READ ROUTINE
433A FE57     CPI      'W'      ;IS IT A WRITE COMMAND?
433C CA9D43    JZ       UWRITE    ;GO TO WRITE ROUTINE
433F C9       RET              ;NOT A GOOD COMMAND-RETURN
4340 F3       UCLEAR: DI        ;DISABLE INTERRUPTS
4341 AF       XRA      A        ;CLEAR A
4342 D3F5     OUT      USCMD     ;OUTPUT THREE TIMES TO ENSURE
4344 D3F5     OUT      USCMD     ;THAT THE USART IS IN A KNOWN STATE
4346 D3F5     OUT      USCMD
4348 3E40     MVI      A,40H     ;CODE TO RESET USART
434A D3F5     OUT      USCMD     ;OUTPUT ON CMD CHANNEL
434C 3E5E     MVI      A,05EH    ;CE IMPLIES ASYN MODE (X16)
;                               ; 8 DATA BITS
;                               ; ODD PARITY
;                               ; 1 STOP BIT
434E D3F5     OUT      USCMD     ;OUTPUT ON CMD CHANNEL
4350 AF       XRA      A        ;CLEAR A, SET ZERO
4351 213942    LXI      H,TCBA   ;CLEAR TCBA AND RCBA
4354 77       MOV      M,A
4355 23       INX      H
4356 77       MOV      M,A
4357 23       INX      H
4358 77       MOV      M,A
4359 23       INX      H
435A 77       MOV      M,A
435B FB       EI              ;ENABLE INTERRUPTS
435C C9       RET              ;AND RETURN TO USER
;
;
;
435D 213B42    UREAD: LXI      H,RCBA ;CHECK READ IDLE
4360 7E       MOV      A,M
4361 B7       ORA      A
4362 C26B43    JNZ      UROUT
4365 23       INX      H
4366 7E       MOV      A,M
4367 B7       ORA      A
4368 CA7743    JZ       URDA      ;READ IS IDLE-PROCEED
436B 3EFE     UROUT: MVI      A,0FEH ;ALREADY RUNNING-ERROR STATUS
436D 217643    LXI      H,URDB   ;SET UP RETURN ADDRESS
4370 E5       PUSH     H         ;PUSH IT INTO STACK
4371 EB       XCHG      ;H GETS COMMAND BLOCK ADDRESS
4372 CD5B42    CALL     CLEAN    ;CALL CLEANUP ROUTINE
4375 E9       PCHL      ;EFFECTIVELY CALLS END ROUTINE
4376 C9       URDB: RET        ;RETURN TO USER
;
4377 EB       URDA: XCHG      ;H GETS COMMAND BLOCK ADDRESS
4378 223B42    SHLD     RCBA     ;RCBA GETS COMMAND BLOCK ADDRESS
437B 3A3742    LDA      LCMD     ;GET LAST COMMAND
437E F616     ORI      16H      ;SET RXE AND DTR AND RESET ERRORS
4380 323742    STA      LCMD     ;AND RETURN TO MEMORY
4383 0F       RRC              ;SET CARRY EQUAL TO TXE
```

APPLICATIONS

4384	D28C43	JNC	URDC	
4387	3E02	MVI	A,2	
4389	323842	STA	TCMD	
438C	07	RLC		
438D	D3F5	OUT	USCMD	;OUTPUT CMD
438F	DBF4	IN	USDAI	;CLEAR USART OF LEFT OVER CHARACTERS
4391	DBF4	IN	USDAI	
4393	3E82	MVI	A,82H	;MDS-CLEAR RECEIVE INTERRUPT
4395	D3F3	OUT	OF3H	;MDS
4397	3EF6	MVI	A,OF6H	;MDS-ENABLE LEVEL THREE
4399	D3FC	OUT	OFCH	;MDS
439B	FB	EI		;ENABLE INTERRUPTS
439C	C9	RET		;RETURN TO USER

439D	213942	UWRITE: LXI	H,TCBA	;CHECK WRITE IDLE
43A0	7E	MOV	A,M	
43A1	B7	ORA	A	
43A2	C26B43	JNZ	UROUT	;BUSY-EXIT
43A5	23	INX	H	
43A6	7E	MOV	A,M	
43A7	C26B43	JNZ	UROUT	;BUSY-EXIT
43AA	EB	XCHG		;OK-H GETS COMMAND BLOCK ADDRESS
43AB	223942	SHLD	TCBA	;TCBA GETS COMMAND BLOCK ADDRESS
43AE	3A3742	LDA	LCMD	;GET LAST COMMAND
43B1	F623	ORI	023H	;SET RTS,DTR, AND TXEN
43B3	323742	STA	LCMD	
43B6	D3F5	OUT	USCMD	
43B8	3EF6	MVI	A,OF6H	;MDS-ENABLE LEVEL THREE INTERRUPTS
43BA	D3FC	OUT	OFCH	;MDS
43BC	FB	EI		;ENABLE SYSTEM INTERRUPTS
43BD	C9	RET		;AND RETURN

APPLICATIONS

```

;*****
;
;      USER IS A TEST PROGRAM WHICH EXERCISES USRUN
;
;*****

43BE 3EC3      USER:  MVI      A,0C3H  ;MDS-SET INTERRUPT VECTOR
43C0 321800    STA      018H
43C3 216842    LXI      H,VECTOR
43C6 221900    SHLD     019H
43C9 3E43      MVI      A,'C'      ;SET GENERAL BLOCK TO A 'C'
43CB 111442    LXI      D,GBLOCK
43CE 12        STAX     D
43CF CD2F43    CALL     USRUN
43D2 210040    LXI      H,BUFIN ;CLEAR INPUT BUFFER
43D5 AF        XRA      A
43D6 77        MOV      M,A
43D7 2C        INR      L
43D8 C2D643    JNZ      $-2
43DB 210041    LXI      H,BUFOUT ;INITIALIZE OUTPUT BUFFER
43DE 75        MOV      M,L
43DF 2C        INR      L
43E0 C2DE43    JNZ      $-2
43E3 65        MOV      H,L      ;REINTIALIZE CONTROL BLOCKS
43E4 2E52      MVI      L,'R'
43E6 220042    SHLD     RBLOCK
43E9 2E57      MVI      L,'W'
43EB 220A42    SHLD     TBLOCK
43EE 6C        MOV      L,H
43EF 220642    SHLD     RCCT
43F2 221042    SHLD     TCCT
43F5 110042    LXI      D,RBLOCK ;START READ
43F8 CD2F43    CALL     USRUN
43FB 110A42    LXI      D,TBLOCK ;START WRITE
43FE CD2F43    CALL     USRUN
4401 3EFF      MVI      A,OFFH ;LOOP WAITING COMPLETION
4403 321642    STA      FLAG      ;FLAG WILL BE SET BY COMPLETION ROUTINES
4406 3A1642    LDA      FLAG
4409 B7        ORA      A
440A C20644    JNZ      $-4
440D 210040    LXI      H,BUFIN ;TEST INPUT BUFFER=OUTPUT BUFFER
4410 7E        COMLP:  MOV     A,M
4411 24        INR      H
4412 BE        CMP      M
4413 C21E44    JNZ      COMER
4416 25        DCR      H
4417 2C        INR      L
4418 C21044    JNZ      COMLP
441B C3BE43    JMP      USER      ;GOOD COMPARE-REPEAT TEST
441E C7        COMER:  RST      0      ;ERROR-RETURN TO MONITOR

0000          END

```

APPLICATIONS

BSTAT 00FF	BUFIN 4000	BUFOU 4100	CEND 0001
CLEAN 425B	COMER 441E	COMLP 4410	EXA 42C1
EXCHA 42BE	FLAG 4216	GBLOC 4214	GSTAT 0000
LCMD 4237	LOADA 423E	MTAB 423D	PEND 42CF
READ 4202	RBLOC 4200	RCBA 423B	RCCT 4206
RCR 4217	RCRA 4208	RISR 4288	RISRA 42AE
RISRB 4299	RISRE 42B9	RRCT 4204	TBAD 420C
TBLOC 420A	TCBA 4239	TCCT 4210	TCMD 4238
TCR 4227	TCRA 4212	TISR 42D4	TISRA 42EC
TRCT 420E	TUTE 4304	TUTE1 4324	TUTE2 4314
TUTE3 431C	TUTE4 4321	UCLEA 4340	URDA 4377
URDB 4376	URDC 438C	UREAD 435D	UROUT 436B
USCMD 00F5	USDAI 00F4	USDAO 00F4	USER 43BE
USRUN 432F	USTAT 00F5	UWRIT 439D	VECTO 4268
VOUT 427E			

APPLICATIONS

APPENDIX A

8251 DESIGN HINTS

1. Output of a command to the USART destroys the integrity of a transmission in progress if timed incorrectly.

Sending a command into the USART will overwrite any character which is stored in the buffer waiting for transfer to the parallel-to-serial converter in the device. This can be avoided by waiting for TxRDY to be asserted before sending a command if transmission is taking place. Due to the internal structure of the USART, it is also possible to disturb the transmission if a command is sent while a SYN character is being generated by the device. (The USART generates a SYN if the software fails to respond to TxRDY.) If this occurrence is possible in a system, commands should be transferred only when a positive-going edge is detected on the TxRDY line.

2. RxE only acts as a mask to RxRDY; it does not control the operation of the receiver.

When the receiver is enabled, it is possible for it to already contain one or two characters. These characters should be read and discarded when the RxE bit is first set. Because of these extraneous characters the proper sequence for gaining synchronization is as follows:

1. Disable interrupts
2. Issue a command to enter hunt mode, clear errors, and enable the receiver (EH,ER,RxE=1)
3. Read USART data (it is not necessary to check status)
4. Enable interrupts

The first RxRDY that occurs after the above sequence will indicate that the SYN character or

characters have been detected and the next character has been assembled and is ready to be read.

3. Loss of CTS or dropping TxEnable will immediately clamp the serial output line.

TxEnable and RTS should remain asserted until the transmission is complete. Note that this implies that not only has the USART completed the transfer of all bits of the last character, but also that they have cleared the modem. A delay of 1 msec following a proper occurrence of TxEmpty is usually sufficient (see item 4). An additional problem can occur in the synchronous mode because the loss of TxEnable clamps the data in at a SPACE instead of the normal MARK. This problem, which does not occur in the asynchronous mode, can be corrected by an external gate combining RTS and the serial output data.

4. Extraneous transitions can occur on TxEmpty while data (including USART generated SYNs) is transferred to the parallel-to-serial converter.

This situation can be avoided by ensuring that TxEmpty occurs during several consecutive status reads before assuming that the transmitter is truly in the empty state.

5. A BREAK (i.e., long space) detected by the receiver results in a string of characters which have framing errors.

If reception is to be continued after a BREAK, care must be taken to ensure that valid data is being received; special care must be taken with the last character perceived during a BREAK, since its value, including any framing error associated with it, is indeterminate.

8251 PROGRAMMABLE COMMUNICATION INTERFACE

